Đã bắt đầu vào lúc	Thứ hai, 24 Tháng mười 2022, 1:13 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ hai, 24 Tháng mười 2022, 2:37 PM
Thời gian thực hiện	1 giờ 24 phút
Điểm	4,00/4,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

Implement method bubbleSort() in class SLinkedList to sort this list in ascending order. After each bubble, we will print out a list to check (using printList).

```
#include <iostream>
#include <sstream>
using namespace std;

template <class T>
class SLinkedList {
public:
    class Node; // Forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    SLinkedList()
    {
      this->head = nullptr;
      this->tail = nullptr;
      this->count = 0;
    }
    ~SLinkedList(){};
    void add(T e)
    {
        Node *pNew = new Node(e);

        if (this->count == 0)
        {
            this->head = this->tail = pNew;
        }
        else
        {
            this->tail->next = pNew;
            this->tail = pNew;
        }

        this->count++;
    }
    int size()
    {
        return this->count;
    }
    void printList()
    {
        stringstream ss;
        ss << "[";
        Node *ptr = head;
        while (ptr != tail)
        {
            ss << ptr->data << ",";
            ptr = ptr->next;
        }

        if (count > 0)
            ss << ptr->data << "]";
        else
            ss << "]";
        cout << ss.str() << endl;
    }
public:
    class Node {
    private:
        T data;
        Node* next;
        friend class SLinkedList<T>;
    public:
        Node() {
```

```
            next = 0;
        }
        Node(T data) {
            this->data = data;
            this->next = nullptr;
        }
    };

    void bubbleSort();
};
```

For example:

Test	Result

int arr[] = {9, 2, 8, 4, 1};
SLinkedList<int> list;
for(int i = 0; i <int(sizeof(arr))/4;i++)
list.add(arr[i]);
list.bubbleSort();
	[2,8,4,1,9]
	[2,4,1,8,9]
	[2,1,4,8,9]
	[1,2,4,8,9]

Answer: (penalty regime: 0 %)

Reset answer

```
template <class T>
void SLinkedList<T>::bubbleSort()
{
    int current = count-1;
    //bool tag = false;
    Node* currentNode = head;
    while(current > 0){
        //tag =  true;
        int step = 0;
        currentNode = head;
        while(step < current){
            if(currentNode->data > currentNode->next->data ){
                T tmp = currentNode->data;
                currentNode->data = currentNode->next->data;
                currentNode->next->data = tmp;
                //tag = false;
            }
            currentNode = currentNode->next;
            step++;
        }
        current--;
        this->printList();
    }
}
```

	Test	Expected	Got	

✔	`int arr[] = {9, 2, 8, 4, 1};`
	`SLinkedList<int> list;`
	`for(int i = 0; i <int(sizeof(arr))/4;i++)`
	`    list.add(arr[i]);`
	`list.bubbleSort();`
		[2,8,4,1,9]
		[2,4,1,8,9]
		[2,1,4,8,9]
		[1,2,4,8,9]
			[2,8,4,1,9]
			[2,4,1,8,9]
			[2,1,4,8,9]
			[1,2,4,8,9]
				✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 2

Chính xác

Điểm 1,00 của 1,00

Implement static method selectionSort in class **Sorting** to sort an array in ascending order. After each selection, we will print out a list to check (using printArray).

```
#include <iostream>
using namespace std;

template <class T>
class Sorting
{
public:
    /* Function to print an array */
    static void printArray(T *start, T *end)
    {
        int size = end - start;
        for (int i = 0; i < size - 1; i++)
            cout << start[i] << ", ";
        cout << start[size - 1];
        cout << endl;
    }

    static void selectionSort(T *start, T *end);
};
```

For example:

Test	Result
int arr[] = {9, 2, 8, 1, 0, -2}; Sorting<int>::selectionSort(&arr[0], &arr[6]);	-2, 2, 8, 1, 0, 9 -2, 0, 8, 1, 2, 9 -2, 0, 1, 8, 2, 9 -2, 0, 1, 2, 8, 9 -2, 0, 1, 2, 8, 9

Answer: (penalty regime: 0 %)

Reset answer

```
1  template <class T>
2  void Sorting<T>::selectionSort(T *start, T *end)
3  {
4      T* current = start;
5          while(current != end-1){
6              T* walker = current+1;
7              T* smallest = current;
8              while(walker != end){
9                  if(*walker < *smallest) smallest = walker;
10                 walker++;
11             }
12             T tmp = *smallest;
13            *smallest = *current;
14             *current = tmp;
15             current++;
16             Sorting<T>::printArray(start,end);
17
18         }
19 }
```

	Test	Expected	Got	

✔	`int arr[] = {9, 2, 8, 1, 0, -2};`
	`Sorting<int>::selectionSort(&arr[0], &arr[6]);`
		`-2, 2, 8, 1, 0, 9`
		`-2, 0, 8, 1, 2, 9`
		`-2, 0, 1, 8, 2, 9`
		`-2, 0, 1, 2, 8, 9`
		`-2, 0, 1, 2, 8, 9`
			`-2, 2, 8, 1, 0, 9`
			`-2, 0, 8, 1, 2, 9`
			`-2, 0, 1, 8, 2, 9`
			`-2, 0, 1, 2, 8, 9`
			`-2, 0, 1, 2, 8, 9`
				✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 3

Chính xác

Điểm 1,00 của 1,00

Implement static methods **sortSegment** and **ShellSort** in class **Sorting** to sort an array in ascending order.

```
#ifndef SORTING_H
#define SORTING_H

#include <sstream>
#include <iostream>
#include <type_traits>
using namespace std;

template <class T>
class Sorting {
private:
    static void printArray(T* start, T* end)
    {
        int size = end - start;
        for (int i = 0; i < size; i++)
            cout << start[i] << " ";
        cout << endl;
    }

public:
    // TODO: Write your code here
    static void sortSegment(T* start, T* end, int segment_idx, int cur_segment_total);
    static void ShellSort(T* start, T* end, int* num_segment_list, int num_phases);
};

#endif /* SORTING_H */
```

For example:

Test	Result
int num_segment_list[] = {1, 3, 5}; int num_phases = 3; int array[] = { 10, 9, 8 , 7 , 6, 5, 4, 3, 2, 1 }; Sorting<int>::ShellSort(&array[0], &array[10], &num_segment_list[0], num_phases);	5 segments: 5 4 3 2 1 10 9 8 7 6 3 segments: 2 1 3 5 4 7 6 8 10 9 1 segments: 1 2 3 4 5 6 7 8 9 10

Answer: (penalty regime: 0 %)

Reset answer

```
1  static void sortSegment(T* start, T* end, int segment, int k) {
2          // TODO
3          int current = segment + k;
4          int size = end - start;
5          while (current < size) {
6              T tmp = start[current];
7              int step = current - k;
8              while (step >= 0 && start[step] > tmp) {
9                  start[step + k] = start[step];
10                 step -= k;
11             }
12             start[step + k] = tmp;
13             current += k;
14             //indexNode += k;
15         }
16     }
17     //template <class T>
18 static void ShellSort(T* start, T* end, int* num_segment_list, int num_phases) {
19     // TODO
```

```
        // Note: You must print out the array after sorting segments to check whether your algorithm is true
    int index = num_phases - 1;
    while (index >= 0) {
        int segment = 0;
        while (segment < num_segment_list[index]) {
            sortSegment(start, end, segment, num_segment_list[index]);
            segment++;
        }
        cout << num_segment_list[index] << " segments: ";
            Sorting<T>::printArray(start, end);
            index--;
    }
}
```

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 4

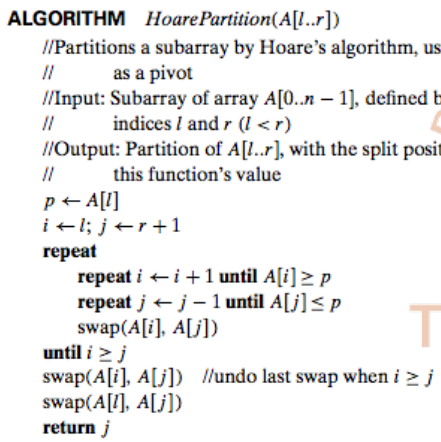
Chính xác

Điểm 1,00 của 1,00

Implement static methods **Partition** and **QuickSort** in class Sorting to sort an array in ascending order.

```
#ifndef SORTING_H
#define SORTING_H
#include <sstream>
#include <iostream>
#include <type_traits>
using namespace std;
template <class T>
class Sorting {
private:
    static T* Partition(T* start, T* end) ;
public:
    static void QuickSort(T* start, T* end) ;
};
#endif /* SORTING_H */
```

You can read the pseudocode of the algorithm used to in method Partition in the below image.



ALGORITHM *HoarePartition*$(A[l..r])$
//Partitions a subarray by Hoare's algorithm, using the first element
// as a pivot
//Input: Subarray of array $A[0..n-1]$, defined by its left and right
// indices l and r $(l < r)$
//Output: Partition of $A[l..r]$, with the split position returned as
// this function's value
$p \leftarrow A[l]$
$i \leftarrow l$; $j \leftarrow r+1$
repeat
  repeat $i \leftarrow i+1$ **until** $A[i] \geq p$
  repeat $j \leftarrow j-1$ **until** $A[j] \leq p$
  swap$(A[i], A[j])$
until $i \geq j$
swap$(A[i], A[j])$ //undo last swap when $i \geq j$
swap$(A[l], A[j])$
return j

For example:

Test	Result

`int array[] = { 3, 5, 7, 10 ,12, 14, 15, 13, 1, 2, 9, 6, 4, 8, 11, 16, 17, 18, 20, 19 };`
`cout << "Index of pivots: ";`
`Sorting<int>::QuickSort(&array[0], &array[20]);`
`cout << "\n";`
`cout << "Array after sorting: ";`
`for (int i : array) cout << i << " ";`
	`Index of pivots: 2 0 0 6 1 0 2 1 0 0 2 1 0 0 0 0 0 0 1 0`
	`Array after sorting: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20`

Answer: (penalty regime: 0 %)

Reset answer

```
1  static T* Partition(T* start, T* end) {
2      // TODO: return the pointer which points to the pivot after rearrange the array.
3      T* i = start+1;
4      T* j = end-1;
5      int index = end-start-1;
6      T tag = *start;
7      do {
8          while(i<=j && *i<=tag) i++;
```

```
        while(i<= j && *j>tag) {
            j--;
            index--;
        }
        if(i<=j){
            T tmp = *i;
            *i = *j;
            *j = tmp;
            i++;
            j--;
            index--;
        }
    } while(i<=j);
    //swap j end start
    T tmp =  *j;
    *j = *start;
    *start = tmp;
    cout<<index<<" ";
    return j;
}

static void QuickSort(T* start, T* end) {
    // TODO
    // In this question, you must print out the index of pivot in subarray after everytime calling method Par
    if(end-start<1) return;
    //cout<<" ";
    T* pivot = Partition(start,end);
    QuickSort(start,pivot);
```

	Test	Expected	Got	

✔	`int array[] = { 3, 5, 7, 10 ,12, 14, 15, 13, 1, 2, 9, 6, 4, 8, 11, 16, 17, 18, 20, 19 };`
	`cout << "Index of pivots: ";`
	`Sorting<int>::QuickSort(&array[0], &array[20]);`
	`cout << "\n";`
	`cout << "Array after sorting: ";`
	`for (int i : array) cout << i << " ";`
		`Index of pivots: 2 0 0 6 1 0 2 1 0 0 2 1 0 0 0 0 0 0 1 0`
		`Array after sorting: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20`
			`Index of pivots: 2 0 0 6 1 0 2 1 0 0 2 1 0 0 0 0 0 0 1 0`
			`Array after sorting: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20`
				✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn